APPENDIX A. Discussion of Configuration Data Used by the Platform

One key feature of the present invention is its ability to have components added to the system and to control the behavior of both existing and new components though configuration data. This appendix describes the structure and content of the configuration data and discusses methods for implementation. Data herein is intended to be illustrative and does not necessarily include all configuration parameters used in any specific embodiment of this invention.

This system organizes configuration data hierarchically, starting with the platform configuration which specifies the names of other components, such as service factories, which in turn may specify alternative services. Each references component in a higher level file may have its own separate configuration.

The system represents configuration data as name-value pairs. Descriptions or comments may also be added. Table 1 provides some of the name value pairs used to represent parameters used by the highest-level platform configuration.

Table 1. System Platform Configuration Data

| Attribute | Value |
|---|---|
| platform.deactivationCycleTime | 600 |
| Platform.serviceDactivationTime | 36000 |
| platform.admin.port | 7070 |
| service.serviceConfigPath | /service |
| service.factories | PlatformSrvs, PowerDemo, TCPSrvs |
| service.directory-service-factory | platformSrvs |
| service.naming-service-factory | platformSrvs |
| service.transport-system-factory | TCPSrvs |
| message.messageConfigPath | /message |
| message.defaultTransport | tcp |
| message.encoders | default, xml |
| message.encoders | default |
| message.handlers | text, assert_revert, kqml |
| message.defaultHandler | assert_revert |
| message.encripters | ssl |
| message.defaultEncripter | none |
| message.interpreterConfigPath | /interpreters |
| inference.interpreters | patternMatcher |
| inference.defaultInterpreter | PatternMatcher |
| ontology.ontologyPath | /ontology |
| ontology.ontologies | PowerPlant |
| ontology.defaultOntology | PowerPlant |

This table illustrates the definition of parameters use to control internal platform behaviors such as activation cycle time (see discussion of FIG. 3), and a TCP port used for remote access to the platform. It also provides a list of service factories and names the service factories used to construct the platform directory, naming and transport services as well as the computer directory path used to locate the configuration files. In a similar manner, this method allows specification of other types of "pluggable" components to support such items as messaging, event handling, and inference and ontology specification as needed. In all cases, this configuration file allows specification of the configuration paths, component names and default selections that the platform may use.

Table 2 presents a service factory configuration for the illustrative embodiment summarized in Appendix B. The service factory configuration specifies the name of the class used to construct the factory (using class-naming conventions used within the Java programming language). It also specifies names for services that the platform will start or connect to when it executes its startup process and service adapters that it will construct.

Table 2. Service Factory Configuration for PowerDemo

| Attribute | Value |
|---|---|
| Service.factory.class | com.power.factory.WarehouseServicesFactory |
| Service.startup | WarehouseERP, CheckPoint |
| Service.adapters | WorkScheduling, WorkOrder |
| WorkOrder | com.power.adapter.WorkOrderAdapter |
| WorkScheduling | com.power.adapter.WorkSchedulingAdapter |

For instance, the data in this table suggests the Warehouse Services Factory will startup two services, one providing platform access the a enterprise systems that tracks tool and parts inventory and schedules work. The other service is monitoring services used to track movement of materials and personnel in and out of the facility and to inform the agent platform of such activity. Specific service adapters enable access to work orders and initiation of work scheduling operations. This configuration file specifies the classes used to construct these services.

Table 3 presents a configuration for a specific message transport service that uses the TCP communications protocol. Here the data specified the class name use to construct the service along with some TCP-related parameters such as the IP address, host name, communications port number and timing attributes.

Table 3. Service Configuration for TCPSrvs.

| Attribute | Value |
|---|---|
| Service.tcp.class | com.agentplaces.platform.message.TCPService |
| service.tcp.hostName | WarehouseServer |
| service.tcp.port | 8080 |
| service.tcp.address | 123.123.123.123 |
| service.tcp.timeout | 1000 |
| service.tcp.linger | 500 |

In one embodiment of this configuration system, the configuration data a set of files containing specification of the form, attribute = value. This format is identical to Property files supported by the Java language. In other embodiments of this configuration method, the system may represent data as serialized object stored within either set of one or more data files or a database. In a further embodiment of this configuration method data is stored as records or relations within a database management system.

APPENDIX B. Background on Illustration Used within User Interface Figures

Many of the figures and some of the discussions concerning this invention cites an illustrative embodiment of this invention that involves using a community of agents to manages the resources needed to maintain the equipment in a power generation plant. This appendix provides some definitional detail on this application, which represents a highly simplified usage scenario.

In general, the illustration envisions three agent platforms representing three facilities within a power generation system;

1. a warehouse and service dispatching system that has responsibility for managing part and tools and dispatching them, alone with service engineers to power generator facilities,

2. an actual power generator facility, and

3. a power distribution substation that has responsibility for requesting power capacity from the generator.

To describe each a table is presented that describe agents and the decisions and interactions. A table is also presented that describe the functions of services installed within or accessible from each platform.

We will consider each separately, starting with the warehouse whose agents and services are shown in Tables B1 and B2.

Table B1. Warehouse Agents.

| Agent | Decisions | Services Used | Agent Communications |
|---|---|---|---|
| Maintenance Scheduler | Request to reschedule, inform scheduled resources | Scheduling (submits constraints and reschedule request). Work Order | Maintenance Engineer, truck, part, tool and Alarm Monitor, Generator Facility Supervisor |
| Logistics Planner | When and what to order | Order processing, parts forecasting, and inventory. | Generator Supervisory Agents concerning power output and Substation Agents concerning power demand |
| Maintenance Engineer | When to report destination, location and capability data | | Checkpoint, Maintenance Scheduling. |
| Maintenance Truck | When to report destination, location, capacity and availability data | | Checkpoint, Maintenance Scheduling |
| Part | When to report destination, location , use and availability data | Inventory | Checkpoint, Maintenance Scheduling |
| Tool | When to report destination, location , use and availability data | Inventory | Checkpoint, Maintenance Scheduling |
| Alarm Monitor | Determines how to handle alarm notifications | Messaging | Checkpoint Agents, Maintenance Scheduler |
| Checkpoint | | Reader and Work Order | Maintenance Engineer, truck, part, tool and alarm. |

Table B2. Warehouse Services

| Service | Description |
|---|---|
| Scheduling | Optimizes resource allocation given constraints |
| Order Processing | Accepts requests to generate orders |
| Work Order | Generates work order data based upon input from the scheduler |
| Inventory | Tracks parts and tools inventory levels |
| Reader | Reads tags, cards or bar codes on people, tools and part entering or existing a facility |
| Parts Forecasting | Generates forecasts for parts |
| Messaging. | Enables the communication of platform events to users via email and/or direct communications with various user display and communication devices |

Within this illustration the Maintenance Scheduling agent has responsibility for optimizing the allocation of materials and personnel to generator facilities serviced by the warehouse facility. It collaborates with agents representing the various maintenance resources (people, parts, etc.) to determine availability of appropriate resources and then setups and uses an optimization service to do the computational work associated with producing a schedule. It may also incorporate scheduling constraints based-upon requests from agents at distribution substations to maintain certain levels of generator capacity. It then informs the appropriate resource and generator facility supervisory agents of the decision. It also initiates work orders using some of the Enterprise Services that are configured for access by the agent platform.

Over time the schedule is executed and monitored. Monitoring activities involve using sensing devices such as bar code readers, magnetic card reader and/or Radio Frequency ID (RFID) tag readers. Reader services generate events that a checkpoint agent can process. This agent can then examine the work order and query the resource agents to verify plan execution. If problems arise, the checkpoint agent issues a message to alarm monitor agent. This alarm monitor then decides what action to take, which could involve notifying the scheduling agent, who may decides to reschedule or notify via messaging services people who can take corrective action.

Tables B3 and B4 suggest some additional agents and a service that would run on the agent platform located at each facility.

In this scenario for the generator facility, a facility supervisor agent would, in anticipation of a maintenance visit, determine the sequence and time of events necessary to shutdown the equipment scheduled for service. This could happen through either agent messaging or by having the agents interact directly with the shutdown service. Equipment agents would also be responsible for diagnosing and report potential equipment failure or problems, perhaps using a alarm monitor agent to decide the appropriate remedy.

The alarm monitor at this facility would operate in a manner similar to its counterpart at the warehouse, although it rules would be customized to comply with policies and personnel at the facility.

Checkpoint agents at the generator serve the same function as within the warehouse, using reader services to monitor movement and reporting deviation from plan and report such deviations to the alarm monitor. These checkpoint agents would utilize remote service access capabilities to access the Enterprise system data (i.e. work orders) and assist maintaining facility security through enforcing access control policies.

Checkpoint agents would also interact with agents defined in table B1 that represent maintenance resources. The checkpoint agents may actually move the resource agents to the platform representing the generator facility upon deployment of the actual physical resources to that location.

Table B2. Warehouse Services

| Service | Description |
|---|---|
| Scheduling | Optimizes resource allocation given constraints |
| Order Processing | Accepts requests to generate orders |
| Work Order | Generates work order data based upon input from the scheduler |
| Inventory | Tracks parts and tools inventory levels |
| Reader | Reads tags, cards or bar codes on people, tools and part entering or existing a facility |
| Parts Forecasting | Generates forecasts for parts |
| Messaging. | Enables the communication of platform events to users via email and/or direct communications with various user display and communication devices |

Within this illustration the Maintenance Scheduling agent has responsibility for optimizing the allocation of materials and personnel to generator facilities serviced by the warehouse facility. It collaborates with agents representing the various maintenance resources (people, parts, etc.) to determine availability of appropriate resources and then setups and uses an optimization service to do the computational work associated with producing a schedule. It may also incorporate scheduling constraints based-upon requests from agents at distribution substations to maintain certain levels of generator capacity. It then informs the appropriate resource and generator facility supervisory agents of the decision. It also initiates work orders using some of the Enterprise Services that are configured for access by the agent platform.

Over time the schedule is executed and monitored. Monitoring activities involve using sensing devices such as bar code readers, magnetic card reader and/or Radio Frequency ID (RFID) tag readers. Reader services generate events that a checkpoint agent can process. This agent can then examine the work order and query the resource agents to verify plan execution. If problems arise, the checkpoint agent issues a message to alarm monitor agent. This alarm monitor then decides what action to take, which could involve notifying the scheduling agent, who may decides to reschedule or notify via messaging services people who can take corrective action.

Tables B3 and B4 suggest some additional agents and a service that would run on the agent platform located at each facility.

In this scenario for the generator facility, a facility supervisor agent would, in anticipation of a maintenance visit, determine the sequence and time of events necessary to shutdown the equipment scheduled for service. This could happen through either agent messaging or by having the agents interact directly with the shutdown service. Equipment agents would also be responsible for diagnosing and report potential equipment failure or problems, perhaps using a alarm monitor agent to decide the appropriate remedy.

The alarm monitor at this facility would operate in a manner similar to its counterpart at the warehouse, although it rules would be customized to comply with policies and personnel at the facility.

Checkpoint agents at the generator serve the same function as within the warehouse, using reader services to monitor movement and reporting deviation from plan and report such deviations to the alarm monitor. These checkpoint agents would utilize remote service access capabilities to access the Enterprise system data (i.e. work orders) and assist maintaining facility security through enforcing access control policies.

Checkpoint agents would also interact with agents defined in table B1 that represent maintenance resources. The checkpoint agents may actually move the resource agents to the platform representing the generator facility upon deployment of the actual physical resources to that location.

Table B3. Generator Facility Agents.

| Agent | Decisions | Services Used | Agent Communications |
|---|---|---|---|
| Facility Supervisor | Report on output, initiate shutdown/ startup processes | Optimal shutdown/ startup | Alarm Monitor, Generator, Transform, Breaker, and Switch. |
| Generator | Diagnosis failures, shutdown/startup, report problem | Optimal shutdown/ startup | Checkpoint, Maintenance Scheduling, Facility Supervisor, Alarm Monitor |
| Transformer | Diagnosis failures, shutdown/startup, report problem | Optimal shutdown/ startup | Checkpoint, Maintenance Scheduling, Facility Supervisor, Alarm Monitor |
| Breaker | Diagnosis failures, shutdown/startup, report problem | Optimal shutdown/ startup | Checkpoint, Maintenance Scheduling, Facility Supervisor, Alarm Monitor |
| Switch | Request to reschedule, inform scheduled resources | Optimal shutdown/ startup | Checkpoint, Maintenance Scheduling, Facility Supervisor, Alarm Monitor |
| Alarm Monitor | Determines how to handle alarm notifications | Messaging | Checkpoint Agents, Facility Supervisor, Alarm Monitor |
| Checkpoint | | Reader and Work Order | Maintenance Engineer, truck, part, tool and alarm. |

Table B4. Generator Facility Services.

| Service | Description |
|---|---|
| Optimal shutdown/ startup | Optimizes the process of shutting down and startup up a generator informing equipment agents when to initiate control actions. |

The supervisor agents at the generator facilities and maintenance-scheduling agents at the warehouse would also need to interact with agents at substations fed by the generators. Tables Br and B6 provide a description of one such agent and a service it might use. To simplify discussion, some decisions that multiple agents at this location might handle are aggregated into this one agent.

The substation supervisor agents primarily role is insuring that substation has sufficient incoming power to meet demands at power load centers. It thus needs to communicate with maintenance scheduling agents in insure that scheduled downtime do not adversely affect its ability to meet demand. The generator supervisors would also communicate unanticipated downtime to the substation supervisor. As a options to changing power generation into the substation, a substation supervisor may request that load centers reduce their power consumption during peak times (load shedding).

A load demand forecasting system is a potentially useful information service for this location. There will also be a set of agents at this facility responsible for diagnosing equipment problems and handling security and maintenance activities an a manner similar to that described earlier for a generator facility.

APPENDIX C. Specifications for and Illustration for Agent Template Language

This appendix provided an embodiment of an agent template language that uses the pattern matcher inference engine. Tables C1 though C8 provide syntax of the language expressed in the standard BNF form. These language elements are illustrative and are not intended to include the specification all features and functions that may comprise alternative embodiments of an agent template language.

In summary, Table C1 provides grammar elements for declaring the agent template as an whole. Table C2 details the references in Table C1 that specifies data tuples or facts. Table C4 specifies the relation element used to define a tuple pattern. Table C4 details the references in Table C1 that specifies rules. Table C6 specifies the portion of the rule that defines predicates. Table C7 expands the condition portion of a predicate. Table C7 specifies the portion of the rule that defines agent tasks.

Table C1. Agent Grammar

| |
|---|
| <start> ::= <agent-stmt> /{agent-function/} |
| <agent-stmt> ::= template-id <start-stmt> /{agent-start-function/} |
| <start-stmt> ::= <fact> /{start-function/} \| <rule> /{default-function/} |

Table C2. Fact Grammar

| |
|---|
| <fact> ::= fact id { <fact-decl> /{fact-function/} |
| <fact-decl> ::= <relation> <fact-inst-list> /{fact-decl-function/} |
| <fact-inst-list> ::= <inst-tuple> <inst-arg-list-tail> /{fact-inst-list-function/} \| <inst-value> <inst-arg-list-tail> /{fact-inst-list-function/} \| } /{fact-inst-list-function/} |
| <inst-tuple> ::= id ( <fact-inst-list> /{inst-tuple-function/} |
| <inst-value> ::= int-value /{inst-value-function/} \| string-value /{inst-value-function/} |
| <inst-arg-list-tail> ::= ,<fact-inst-list> /{inst-arg-list-tail-function/} \| ) /{inst-arg-list-tail-function/} \| } /{inst-arg-list-tail-function/} |

Table C4. Relation Grammar

| |
|---|
| <relation> ::= relation id ( <relation-arg-list> /{relation-function/} |
| <relation-arg-list> ::= <relation-tuple> <relation-arg-list-tail> /{relation-arg-list-function/} \| <variable-arg> <relation-arg-list-tail> /{relation-arg-list-function/} |
| <relation-tuple> ::= id ( <relation-arg-list> /{relation-tuple-function/} |
| <variable-arg> ::= <type-specifier> variable-id /{variable-arg-function/} |
| <type-specifier> ::= int /{type-specifier-function/} \| string /{type-specifier-function/} |
| <relation-arg-list-tail> ::= , <relation-arg-list> /{relation-arg-list-tail-function/} \| ) /{relation-arg-list-tail-function/} |

Table C5. Rule Grammar

| |
|---|
| <rule> ::= rule id { <relation-list> <predicate-stmt> <condition> } /{rule-function/} |
| <relation-list> ::= <relation> <relation-list-tail> /{relation-list-function/} |
| <relation-list-tail> ::= & <relation> <relation-list-tail> /{relation-list-tail-function/} \| ; /{default-function/} |

Table C6. Predicate Grammar

<predicate-stmt> ::= predicate <predicate-assgn-stmt> <predicate-assgn-list> /{predicate-stmt-function/}

<predicate-assgn-stmt> ::= variable-id = <predicate-expr> ; /{predicate-assgn-stmt-function/}

<predicate-assgn-list> ::= <predicate-assgn-stmt> <predicate-assgn-list> /{default-function/} | epsilon /{default-function/}

<predicate-expr> ::= <predicate-term> <predicate-expr'> /{predicate-expr-function/}

<predicate-term> ::= <predicate-factor> <predicate-term'> /{predicate-term-function/}

<predicate-factor> ::= ( <predicate-expr> ) /{predicate-factor-function/} | variable-id /{predicate-factor-function/} | int-value /{predicate-factor-function/} | string-value /{predicate-factor-function/}

<predicate-term'> ::= <rel-opr> <predicate-factor> /{predicate-term'-function/} | epsilon /{predicate-term'-function/}

<predicate-expr'> ::= <log-opr> <predicate-term> <predicate-expr'> /{predicate-expr'-function/} | epsilon /{predicate-expr'-function/}

<predicate-expr-list> ::= <predicate-expr> ; <predicate-expr-list> /{predicate-expr-list-function/} | epsilon /{predicate-expr-list-function/}

<rel-opr> ::= <= /{rel-opr-function/} | < /{rel-opr-function/} | == /{rel-opr-function/}| > /{rel-opr-function/} | >= /{rel-opr-function/} | != /{rel-opr-function/}

<log-opr> ::= && /{log-opr-function/} | || /{log-opr-function/}

Table C7. Condition Grammar

<condition> ::= condition <condition-label> { <lhs-stmt> <rhs-stmt> } /{default-function/}

<condition-label> ::= id /{default-function/}| epsilon /{default-function/}

<lhs-stmt> ::= lhs < predicate-expr> ; /{default-function/}

<rhs-stmt> ::= rhs <task-decl> /{default-function/}

Table C8. Task Grammar

<task-decl> ::= <service-call> <task-decl-tail> /{default-function/}

<task-decl-tail> ::= , <task-decl> /{default-function/} | ; /{default-function/}

<service-call> ::= id ( <service-arg-list> /{default-function/}

< service -arg-list> ::= < service -call> < service -arg-list-tail> /{default-function/} | variable-id < service -arg-list-tail> /{default-function/}

< service -arg-list-tail> ::= , < service -arg-list> /{default-function/} | ) /{default-function/}